Direct-List Algorithm for Configuration Interaction Calculations

LAURA GAGLIARDI, GIAN LUIGI BENDAZZOLI, STEFANO EVANGELISTI

Dipartimento di Chimica Fisica e Inorganica, Università di Bologna, Viale Risorgimento 4, 40136 Bologna, Italy

Received 8 October 1996; accepted 25 January 1997

ABSTRACT: We present a method for the direct generation of the lists of strings, suited for integral-driven full-CI (FCI) algorithms. This method generates the string lists each time they are used, and hence sensibly reduces the memory requirements, compared to our previous method that precalculates the lists. It was also extended to permit a truncation of the string space, according to the level of excitation. © 1997 by John Wiley & Sons, Inc. *J Comput Chem* **18:** 1329–1343, 1997

Introduction

In configuration interaction (CI) algorithms, the action of the Hamiltonian on the wave function can be computed once it is known the effect of excitation operators on the strings that compose the Slater determinants of the system. This is usually done by precomputing the effect of one-electron (and sometimes also two-electron) excitations on the set of all the strings. There are several full-CI (FCI) algorithms based on this approach.[1–12] In this article, we illustrate a method for the direct generation of these lists of strings, that is suited for integral-driven FCI algorithms, and was implemented in our FCI[12] code, both in the scalar/vector versions (RISC IBM, Cray C90) and in the parallel version (Cray T3D).[13,14] It was also extended to permit a truncation of the string space, according to the level of excitation.

This method calculates the string lists each time they are needed, in order to avoid all the problems connected with their storage. With our original algorithm,[12] in which the arrays containing both the monoelectronic and bielectronic lists were precalculated and stored in memory or on disk, it was unfeasible to treat systems with a large number of orbitals, especially when the number of electrons is far from half-filling. This is indeed the case in most molecular computations, where extended atomic orbital (AO) bases are required to get good accuracy. In fact, in these cases, the dimensions of the string lists soon became large compared to the CI vector, and it was impossible to store all of them in memory. With the present method, on the other hand, we bypassed the problems associated

Correspondence to: L. Gagliardi

Contract/grant sponsor: EEC Human Capital Mobility Program; contract/grant number: ERBCHRXCT96/0086

CCC 0192-8651 / 97 / 111329-15

with the list storage and memory requirements were drastically reduced. At the same time, the code did not lose its efficiency, so it was possible to study systems that were dimensionally prohibitive with the previous version of the code.

In the following section we briefly resume the CI formalism used in our FCI algorithm. In the next two sections the direct generation of the lists of strings is presented. Then we describe the way in which we truncate the string space, for the cases in which it is impossible to include all the strings in the calculations. Finally, in the last section, we report some test results obtained with the present direct-list method.

Direct CI Formalism

The time-independent Shrödinger equation for a multielectron system reads:

$$\hat{H}|\Psi_{ex}\rangle = E_{ex}|\Psi_{ex}\rangle \tag{1}$$

where $\hat{H}$ is the Hamiltonian operator of the system, $[\Psi_{ex}\rangle$ is an exact eigenstate, and E_{ex} is the associated total electronic energy. In practice, it is not possible, except in special cases, to solve eq. (1) analytically, because the wave function describes the correlated motion of the interacting particles and approximations must be introduced. In the CI method an approximate solution, $|\Psi\rangle$, to eq. (1) is obtained by expanding the wave function as a finite linear combination of Slater determinants, $|\Phi_R\rangle$. These determinants are constructed from a set of orthonormal orbitals, usually the molecular orbitals (MO) of the system:

$$|\Psi\rangle = \sum_R X_R|\Phi_R\rangle \tag{2}$$

where

$$X_R = \langle\Phi_R|\Psi\rangle \tag{3}$$

is the component of a vector $\mathbf{X}$ of the expansion coefficients. In a similar way, the Hamiltonian operator can be expanded in the Slater determinant basis:

$$\hat{H} = \sum_{RS}|\Phi_R\rangle H_{RS}\langle\Phi_S| \tag{4}$$

where

$$H_{RS} = \langle\Phi_R|\hat{H}|\Phi_S\rangle \tag{5}$$

is an element of the Hamiltonian matrix $\mathbf{H}$. Eq. (1) becomes thus a matrix eigenvalue equation

$$\mathbf{HX} = E\mathbf{X} \tag{6}$$

In principle, the computation of the CI wave function could be carried out by diagonalizing the Hamiltonian matrix but for large CI expansions containing millions of determinants it is impossible to diagonalize matrices of this size using standard techniques. Usually, however, a complete diagonalization is neither required nor desirable, because we are interested only in a few of the lowest roots, which can be determined by iterative methods.[15,16] In these methods, the eigenvectors are determined through a sequence of linear transformations of the form:

$$\mathbf{Y} = \mathbf{HX} \tag{7}$$

where $\mathbf{X}$ is some trial vector. This fundamental operation can be implemented in the trivial way by computing and storing the elements of $\mathbf{H}$, or directly from the one- and two-electron integral lists, as it is currently done in the so-called direct CI methods.[17,18]

Each Slater determinant of the CI expansion [eq. (2)] can be written as the tensor product of two strings, one of the occupied alpha orbitals, σ_α, and one of the occupied beta orbitals, σ_β[1]:

$$|\Phi\rangle = |\sigma_\alpha\sigma_\beta\rangle \tag{8}$$

A string is an ordered product of creation operators acting on the vacuum, and can be represented as an ordered sequence of occupation numbers, 1 for occupied and 0 for empty orbitals. Therefore, for a system containing m_α (α type) and m_β (β type) electrons within n orbitals, each string is represented by a binary word of length n, containing m 1's and $(n - m)$ 0's, where m is either m_α or m_β. This representation is particularly convenient on a computer, because it is possible to operate on the strings by performing bit operations on binary numbers.

The strings composed of m 1's and $(n - m)$ 0's are a standard representation of the combinations of n objects, m at a time, so we can exploit well-known algorithms to generate all the strings in lexical order.[19] For sake of simplicity, in the following discussion, we will not take into account the spatial symmetry of the system. We denote by $\mathscr{L}$ the list of the strings in lexical order, whose length is:

$$\mathscr{N} = \binom{n}{m} \tag{9}$$

Each string σ of the list is identified by its address (I), given simply by its order of generation (I_α is the address of σ_α, and I_β of σ_β). The wave function $|\Psi\rangle$ in terms of strings becomes:

$$|\Psi\rangle = \sum_{\sigma_\alpha,\,\sigma_\beta} X_{I_\alpha I_\beta} |\sigma_\alpha \sigma_\beta\rangle \tag{10}$$

with $\hat{H} = \Psi\rangle$ given by:

$$\hat{H} = \Psi\rangle = \sum_{\tau_\alpha,\,\tau_\beta} Y_{J_\alpha J_\beta} |\tau_\alpha \tau_\beta\rangle \tag{11}$$

where $J_\alpha(J_\beta)$ is the address of the τ_α (τ_β) string. The Hamiltonian in the second quantization formalism reads:

$$\hat{H} = \frac{1}{2} \sum_{\mu,\,\nu=\alpha,\,\beta} \sum_{i,\,j,\,k,\,l} \langle kl|ij\rangle \hat{E}_{ij}^{kl}(\mu, \nu) \tag{12}$$

with:

$$\hat{E}_{ij}^{kl}(\mu, \nu) = a_{k\mu}^{+} a_{l\nu}^{+} a_{j\nu} a_{i\mu} \tag{13}$$

The greek indices (μ, ν) refer to spin and i, j, k, l are the orbital indices. In this notation we have included the monoelectronic part of the Hamiltonian into the bielectronic one, by redefining the two-electron matrix element according to:

$$\langle ij|kl\rangle = (ik|jl) + \frac{1}{m_\alpha + m_\rho - 1}(h_{ik}\delta_{jl} + h_{jl}\delta_{ik}) \tag{14}$$

where h_{ij} and $(ik|jl)$ are the usual one- and two-electron integrals.

The Hamiltonian contains four different terms according to the spin:

$$\hat{H} = \hat{H}_{\alpha\alpha} + \hat{H}_{\alpha\beta} + \hat{H}_{\beta\alpha} + \hat{H}_{\beta\beta} \tag{15}$$

In the following we will first consider the contribution of $\hat{H}_{\beta\beta}$ to the vector **Y**. The action of $\hat{H}_{\alpha\alpha}$ is similar, and, in case $m_\alpha = m_\beta$, it can be obtained from that of $\hat{H}_{\beta\beta}$, by simple transposition, as discussed in Ref. 5. We will then consider the contribution of $\hat{H}_{\alpha\beta}$.

$\mathbf{H_{BB}}$

The operator $\hat{E}_{ij}^{kl}(\beta)$ acts only on the σ_β strings giving a result of zero if the orbitals i or j are not occupied, or if k or l are occupied in the string σ_β. Otherwise, it gives a new string, τ_β, multiplied by a phase factor $(-1)^p$. The phase factor is given by the parity of the permutation, needed to reorder the orbitals of τ_β in increasing order. For each quadruple i, j, k, l we can construct the list $\mathscr{L}_{ij}^{kl}$ of the strings having i, j occupied and k, l virtual, and the list $\mathscr{L}_{kl}^{ij}$ of the strings with k, l occupied and i, j virtual. Because $\hat{E}_{ij}^{kl}(\beta) = -\hat{E}_{ji}^{kl}(\beta)$, we assume $i > j$, and, for the same reason, $k > l$. The length of $\mathscr{L}_{ij}^{kl}$ and $\mathscr{L}_{kl}^{ij}$ is:

$$\mathscr{N}_{ij}^{kl} = \mathscr{N}_{kl}^{ij} = \binom{n-4}{m-2} \tag{16}$$

if the four indices are different, which is by far the most frequent case. If, on the other hand, the pair i, j is equal to the pair k, l, we have:

$$\mathscr{N}_{ij}^{ij} = \binom{n-2}{m-2} \tag{17}$$

The list, $\mathscr{L}_{ij}^{kl}$, contains all the strings of the general list $\mathscr{L}$ on which the operator $\hat{E}_{ij}^{kl}$ can act without giving zero. The result of operator $\hat{E}_{ij}^{kl}$ acting on the strings of $\mathscr{L}_{ij}^{kl}$ are the strings of the list $\mathscr{L}_{kl}^{ij}$. Because in each list the strings are in lexical order, the effect of applying $\hat{E}_{ij}^{kl}$ to the rth element of $\mathscr{L}_{ij}^{kl}$ is to produce the rth element of $\mathscr{L}_{kl}^{ij}$:

$$\hat{E}_{ij}^{kl}\mathscr{L}_{ij}^{kl} = \mathscr{S}_{kl}^{ij}\mathscr{L}_{kl}^{ij} \tag{18}$$

where we indicated with $\mathscr{S}_{kl}^{ij}$ a list of the parity factors of the strings of $\mathscr{L}_{kl}^{ij}$, and the product $\mathscr{S}_{kl}^{ij}\mathscr{L}_{kl}^{ij}$ in eq. (18) is done element by element. Eq. (18) holds because the lexical order is determined only by the bits other than b_i, b_j and b_k, b_l with the latter pair being fixed within each list. If bits b_i, b_j and b_k, b_l are suppressed in both lists we will have two lists of all the strings of length $(n-4)$ representing the combinations of $(n-4)$ items $(m-2)$ at a time, in lexical order, which are perfectly coincident.

Let us now consider the effect of $\hat{H}_{\beta\beta}$ on the wave function:

$$\begin{aligned}\hat{H}_{\beta\beta}|\Psi\rangle &= \frac{1}{2}\sum_{ijkl}\langle kl|ij\rangle \hat{E}_{ij}^{kl}(\beta)|\Psi\rangle \\ &= \frac{1}{2}\sum_{ijkl}\langle kl|ij\rangle \sum_{\sigma_\alpha,\,\sigma_\beta} X_{I_\alpha I_\beta}\hat{E}_{ij}^{kl}(\beta)|\sigma_\alpha\sigma_\beta\rangle \\ &= \frac{1}{2}\sum_{ijkl}\langle kl|ij\rangle \sum_{\sigma_\alpha,\,\tau\beta} X_{I_\alpha I_\beta}|\sigma_\alpha\tau_\beta\rangle\end{aligned} \tag{19}$$

If we introduce eq. (11) into eq. (19), we observe that the element $Y_{I_\alpha J_\beta}$ of **Y** is updated with $\langle kl|ij\rangle X_{I_\alpha I_\beta}$. Therefore, the action of $\hat{H}_{\beta\beta}$ for each contribution of the integral $\langle kl|ij\rangle$, can be computed once the lists of the addresses associated to $\mathscr{L}_{ij}^{kl}$ and $\mathscr{L}_{kl}^{ij}$ are known. If we indicate, by L_{ij}^{kl} and

L_{kl}^{ij}, these two lists of addresses, this contribution is computed as follows:

loop $I = 1, \mathcal{N}_{ij}^{kl}(\beta)$

$I_\beta = L_{ij}^{kl}(I)$

$J_\beta = L_{kl}^{ij}(I)$

loop $J = 1, \mathcal{N}(\alpha)$

$Y(J, J_\beta) = Y(J, J_\beta) + \langle ij|kl\rangle X(J, I_\beta)$

end loop J

end loop I

$H_{\alpha\beta}$

To investigate the effect of the component $\hat{H}_{\alpha\beta}$ of the Hamiltonian on the wave function, for each couple of orbitals i, k, we consider the monoelectronic lists $\mathcal{L}_i^k$ of the strings with i occupied and k virtual, and the list $\mathcal{L}_k^i$ of the strings with k occupied and i virtual. The length of these two lists is:

$$\mathcal{N}_i^k = \mathcal{N}_k^i = \binom{n-2}{m-1} \tag{20}$$

if i is different from k, and:

$$\mathcal{N}_i^i = \binom{n-1}{m-1} \tag{21}$$

if the two indices are coincident. Similar to the bielectronic case, the action of the monoelectronic operator $\hat{E}_i^k(\mu)(a_{k\mu}^+ a_{i\mu})$ on the strings of the $\mathcal{L}_i^k$ list gives the strings of the $\mathcal{L}_k^i$ list, and the one-to-one correspondence between the elements of the two lists is preserved:

$$\hat{E}_i^k \mathcal{L}_i^k = \mathcal{S}_k^i \mathcal{L}_k^i \tag{22}$$

Let us now consider the effect of $\hat{H}_{\alpha\beta}$ onto the wave function:

$$\begin{aligned}\hat{H}_{\alpha\beta}|\Psi\rangle &= \sum_{ijkl}\langle kl|ij\rangle \hat{E}_i^k(\alpha)\hat{E}_j^l(\beta)|\Psi\rangle = \\ &= \sum_{ijkl}\langle kl|ij\rangle \sum_{\sigma_\alpha,\sigma_\beta} X_{I_\alpha I_\beta}\hat{E}_i^k(\alpha)\hat{E}_j^l(\beta)|\sigma_\alpha\sigma_\beta\rangle = \\ &= \sum_{ijkl}\langle kl|ij\rangle \sum_{\tau_\alpha,\tau_\beta} X_{I_\alpha I_\beta}|\tau_\alpha\tau_\beta\rangle \end{aligned} \tag{23}$$

As in the bielectric case, we generate the list of the string addresses, L_i^k, for each i, k pair. The $Y_{J_\alpha J_\beta}$ element of $\mathbf{Y}$ is updated as follows for each contribution of the integral $\langle ij|kl\rangle$:

loop $I = 1, \mathcal{N}_i^k(\beta)$

$I_\beta = L_i^k(I)$

$J_\beta = L_k^i(I)$

loop $J = 1, \mathcal{N}_j^l(\alpha)$

$I_\alpha = L_j^l(J)$

$J_\alpha = L_l^j(J)$

$Y(J_\alpha, J_\beta) = Y(J_\alpha, J_\beta) + \langle ij|kl\rangle X(I_\alpha, I_\beta)$

end loop J

end loop I.

List Generation

In the original version of the algorithm[12] the arrays containing the lists were precalculated and stored in memory or on disk. In the following the method for the direct generation of string lists is described. Suppose we want to generate the monoelectronic list of addresses L_i^k. We consider the global list of strings, $\mathcal{L}$, select those strings having i occupied and k virtual, and include their address in the list L_i^k.

This is done by building a mask (`mask_ik`); that is, a binary number with bit equal to 1 in positions i and k and 0 in all the other positions, and a second mask (`mask_i`) with 1 in position i and 0 in all other positions. We then loop over the strings of $\mathcal{L}$ and test if the *logical and* of each string with `mask_ik` is equal to `mask_i`. When a string satisfies this requirement we store its address in the array of the addresses and increment a counter, to determine the length of the array. In the FORTRAN implementation, to set a bit equal to 1 we use the intrinsic function `IBSET`, whereas to do the *logical and* of each string and `mask_ik` we use the intrinsic function `AND`.

For each pair i, k, we thus have to build one array containing the string addresses, and one for the phase factors. The address of a string is given by its generation order in the list $\mathscr{L}$, and the phase factor is $(-1)^p$. This is determined by the phase factors associated with the two operators a_i^+, a_k, respectively, given by $(-1)^{p_i}$ and $(-1)^{p_k}$. Here, p_i, p_k are the number of transpositions needed to restore the increasing order of the occupied orbitals when we act with the operators a_i^+ and a_k on a string. The total parity of the string will thus be:

$$(-1)^p = (-1)^{p_i+p_k} = (-1)^{(2p_i)+(p_k-p_i)} = (-1)^{p_k-p_i} \tag{24}$$

which is simply given by the number of occupied orbitals between i and k.

On the computer this is realized by building a mask, `mask_s`, with ones in all the positions between i and k. The *logical and* of `mask_s` and the string in exam, `l-string`, is then done, and we finally count the number of bits equal to one in the resulting string, `AND(mask_s,l_string)`. The parity factor is $(+1)$ if this number is even, and (-1) if the number is odd. To count the bits equal to one in a string, we used the function `POPCNT`, which on some computers, like the Cray C90 or T3D is a library function, whereas in the case of the RISC 6000 it was written in C language. In the following we report some sections of the FORTRAN code used to generate the arrays of the lists and the parity factors. With our notation, the orbitals are labeled from 1 to n, whereas the bits, in a computer word, from 0 to $(n-1)$. Hence, the orbital i corresponds to the bit $(i-1)$. Furthermore, for systems involving a large number of orbitals, our formalism can be extended straight to the case of strings stored in more than one word:

```
! Construction of the lists L_ik = string addresses
!                           S_ik = phase phactors
! Remember that orbital i corresponds to bit i-1, etc.

! prepare mask_i  = mask with 1 in position i
!         mask_ik = mask with 1 in positions i and j

   mask_i  = IBSET (0, i-1)
   mask_ik = IBSET (maski, k-1)

! prepare mask_s = mask with 1 in the positions i<1<k

   mask_s = 0
   DO l = i+1, k-1
     mask_s = IBSET (mask_s, l-1)
   ENDDO

! loop over all the strings, array l_string
!
! compute N_ik = counter of the list length
!         L_ik = array of the addresses
!         S_ik = array of the phase factors

   N_ik = 0
   DO ii = 1, N
      IF (AND(mask_ik,l_string(ii)) .EQ. mask_i) THEN
         N_ik = N_ik+1
         L_ik(N_ik) = ii
         S_ik(N_ik) = (-1) ** POPCNT (AND(mask_s,l_string(ii)))
         ENDIF
   ENDDO
!
!
```

Here, we report the C version of the POPCNT function.

```
!
! POPCNT function in C language
!
INT POPCNT (x)
UNSIGNED *x;
{
    UNSIGNED localx;
    INT b;
    localx = *x;
    FOR (b=0; localx !=0 ; localx >>=1)
        IF (localx & 01) b++;
    RETURN b;
}

!
!
```

In a similar way we generate the list, L_k^i, of the addresses of the strings with k occupied and i virtual. We do not need to determine the string parity factor, because, for each string, it is the same as the one for the corresponding string in the L_i^k list. For the bielectronic lists we proceed in a similar way. In this case, however, the parity factor is determined as the product of two monoelectronic parity factors. If, for example, we want to calculate the parity factor of a string belonging to the list L_{ij}^{kl}, we count the number of bits equal to one between i and l, and those between j and k. The total parity factor is given by the product of these two numbers.

Memory and CPU Requirements

The direct generation of string lists is an appealing method, because it implies a substantial savings in the use of the computer memory or disk. To consider it of greatest convenience, however, it is necessary that the overhead of CPU time required to produce the lists at each iteration is small with respect to the total CPU time required by the CI process itself. This happens when the strings contain a large number of occupied orbitals, because in this case the number of the strings included in the lists is almost of the same order of magnitude of the total number of strings examined.

When, on the other hand, we treat strings with a small number of occupied orbitals (three or less electrons), the length of the mono- and bielectronic lists is short compared to the number of total strings. Therefore, the time spent for the list generation can become a relevant fraction of the time needed to the CI iteration. The ratio between the selected and examined strings is, in fact, given by:

$$\mathcal{N}_i^j/\mathcal{N} = \frac{(n-m)m}{n(n-1)} \tag{25}$$

For m small compared to n, this ratio is m/n. For the two-electron lists the situation is even more dramatic, because the ratio between the selected and examined strings is:

$$\mathcal{N}_{ij}^{kl}/\mathcal{N} = \frac{(n-m)(n-m-1)m(m-1)}{n(n-1)(n-2)(n-3)} \tag{26}$$

which goes as $m(m-1)/n^2$ for $m \ll n$.

In these cases, instead of directly generating all the lists, it is more convenient to preliminarily generate and store some intermediate lists and directly generate the required final lists. At the beginning of the CI procedure, we precalculate and store in memory all the lists L_i^i of the strings with i occupied. We then generate, with the direct algorithm, the final list, L_i^j, each time it is needed. In doing that, it is sufficient to make the selection among the strings pointed by L_i^i, instead of scanning all the strings of the global list $\mathcal{L}$.

The success of this mixed approach can be easily explained if we note that the ratio, $\mathcal{N}_i^j/\mathcal{N}_i^i$ [see eqs. (20) and (21)], is given by:

$$\mathcal{N}_i^j/\mathcal{N}_i^i = \frac{n-m}{n-1} \tag{27}$$

In the bielectronic case, we precalculate the list L_{ij}^{ij} with i and j occupied, and then directly generate the list L_{ij}^{kl}, examining the strings of the list L_{ij}^{ij}. The ratio $\mathcal{N}_{ij}^{kl}/\mathcal{N}_{ij}^{ij}$ [see eqs. (16) and (17)] is given by:

$$\mathcal{N}_{ij}^{kl}/\mathcal{N}_{ij}^{ij} = \frac{(n-m)(n-m-1)}{(n-2)(n-3)} \tag{28}$$

For small m, the two quantities, eqs. (27) and (28), become close to 1, which means that almost all the examined strings of L_i^i, L_{ij}^{kl} are included in the final lists L_i^j, L_{ij}^{ij}, respectively.

From now on we will refer to method that precalculates the intermediate lists and directly generates only the final ones, as the two-step method, whereas the one with the direct generation of all the lists in their final form as the one-step method. By making use of the relations

that define the length of the different types of lists, one can estimate the memory (or disk) requirements in the cases of precomputation, two-step, and one-step direct generation of the string lists. The results are summarized in Table I. We notice that, within the one-step method, there is the lowest need of memory sources, followed by the two-step one and finally the precomputation.

In a similar way, we can estimate the CPU time needed, at each iteration, for the generation of the lists. In Table II the operation count for the generation of mono- and bielectronic lists and CI multiplication in the three different cases (precomputation, two-step, one-step) are reported. The choice of the method must be a compromise between the list-generation time and the memory requirements. In general, the two-step method is more convenient than the one-step, but sometimes, if the arrays precalculated and stored become extremely large, it is unfeasible to use the two-step method, and the one-step approach must thus be followed. This is the case when some truncation in the string space is performed, which will be described in the next section.

Truncation of String Space

When a FCI is not practicable because of dimensional problems, a truncation of the CI space is needed. One popular truncation scheme is to excite up to a certain level from a CAS space. If one wants to work with a truncated CI (TCI) space it is necessary to be able to produce lists of strings in a truncated space. We thus implemented a truncation scheme based on the truncation of the space of the strings.

Let us partition the n orbitals into three classes, occupied **O**, active **A**, and virtual **V**. Occupied orbitals are always occupied in the reference strings, virtual orbitals are always empty, and active orbitals can be either occupied or empty. We denote by n_o, n_a, and n_v the number of orbitals in the three classes, and by m_o, m_a, and m_v the corresponding number of electrons. Therefore, the reference strings are characterized by the relations:

$$m_o = n_o \tag{29}$$

and;

$$m_v = 0 \tag{30}$$

whereas, for *all* the strings, we have, obviously:

$$m_o + m_a + m_v = m \tag{31}$$

A string will have, in general, p holes in the occupied orbitals and q electrons in the virtual orbitals. Because the total number of electrons, m, is fixed, the number of electrons in the active orbitals is automatically determined as $m_a = m - n_o + p - q$. We indicate, by $\mathscr{L}[p, q]$, the set of strings with, at most, p holes in the occupied orbitals, and q virtual electrons.

Suppose we want to work with strings with, at most, p_{max} holes in the occupied orbitals and q_{max} electrons in the virtual orbitals. The problem is now that, in general, an excitation operator, $\hat{E}_i^k$, can transform a string that belongs to $\mathscr{L}[p, q]$ into a string not belonging to $\mathscr{L}[p, q]$. Therefore, the list generation cannot be done by simply replacing $\mathscr{L}[p_{max}, q_{max}]$ with $\mathscr{L}$ in the formulae of ''Direct CI Formalism'' but we have to test if the strings

TABLE I.
Memory Requirements for Monoelectronic- and Bielectronic-List Generation, and FCI Multiplication within the Three Methods: Precomputation and Two-Step, and One-Step Methods.[a]

List Generation	Precomputation	Two Steps	One Step
One-electron lists	$n^2\binom{n-2}{m-1}$	$n\binom{n-1}{m-1}$	$\binom{n}{m}$
Two-electron lists	$\binom{n}{2}^2\binom{n-4}{m-2}$	$\binom{n}{2}\binom{n-2}{m-2}$	$\binom{n}{m}$
FCI vector	$\binom{n}{m}^2$	$\binom{n}{m}^2$	$\binom{n}{m}^2$

[a] n is the number of orbitals and m is the number of electrons of a given spin.

TABLE II.
Operation Count (for each Quadruple *i*, *j*, *k*, *l*) for Monoelectronic-and Bielectronic-List Generation, and FCI Multiplication within the Three Methods:Precomputation and Two-Step and One-Step Methods.[a]

List Generation	Precomputation	Two Steps	One Step
One-electron lists	0	$\binom{n-1}{m-1}$	$\binom{n}{m}$
Two-electron lists	0	$\binom{n-2}{m-2}$	$\binom{n}{m}$
$FCI_{\alpha\beta}$	$\binom{n-2}{m-1}^2$	$\binom{n-2}{m-1}^2$	$\binom{n-2}{m-1}^2$
$FCI_{\beta\beta}$	$\binom{n-4}{m-2}\binom{n}{m}$	$\binom{n-4}{m-2}\binom{n}{m}$	$\binom{n-4}{m-2}\binom{n}{m}$

[a] n is the number of orbitals and m is the number of electrons. No spatial symmetry is taken into account.

obtained by the effect of the excitation operator acting on the strings of $\mathscr{L}[p, q]$ still belong to the CI space and, in case they do not, discard them. The effect of a general creation (a_i^+) and annihilation operator (a_i) on the $\mathscr{L}[p, q]$ lists is:

i	$a_i^+\mathscr{L}[p, q]$	$a_i\mathscr{L}[p, q]$
O	$\mathscr{L}[p-1, q]$	$\mathscr{L}[p+1, q]$
A	$\mathscr{L}[p, q]$	$\mathscr{L}[p, q]$
V	$\mathscr{L}[p, q+1]$	$\mathscr{L}[p, q-1]$

We observe that the creation operator, a_i^+, decreases the first index, p, by 1 when i is an **O** orbital, whereas the annihilator operator, a_i, increases p by 1 when i is an **O** orbital. Both a_i^+ and a_i leave the two indices p and q invariant when i is an **A** orbital. Finally, a_i^+ increases q by 1 and a_i decreases q by 1 when i is a **V** orbital.

Therefore eqs. (18) and (22) are replaced by:

$$\hat{E}_{ij}^{kl}\mathscr{L}_{ij}^{kl}[p, q] = \mathscr{S}_{kl}^{ij}\mathscr{L}_{kl}^{ij}[p + \delta_p, q + \delta_q] \quad (32)$$

and:

$$\hat{E}_i^k\mathscr{L}_i^k[p, q] = \mathscr{S}_k^i\mathscr{L}_k^i[p + \delta_p, q + \delta_q] \quad (33)$$

Here δ_p and δ_q are given by:

$$\delta_p = \text{number of annihilated } \mathbf{O} \text{ orbitals} - \text{number of created } \mathbf{O} \text{ orbitals}$$

and:

$$\delta_q = \text{number of created } \mathbf{V} \text{ orbitals} - \text{number of annihilated } \mathbf{V} \text{ orbitals}$$

When we make the selection of the strings to build the list $\mathscr{L}_i^k$, we must be sure that not only is each string σ that we select an acceptable string (this is simply done by running over the strings of $\mathscr{L}[p_{max}, q_{max}]$), but that $\hat{E}_i^k\sigma$ is also an acceptable string. So, we must have:

$$0 \le p + \delta_p \le p_{max} \quad (34)$$

and

$$0 \le q + \delta_q \le q_{max} \quad (35)$$

from which we obtain:

$$-\delta_p \le p \le p_{max} - \delta_p \quad (36)$$

$$-\delta_q \le q \le q_{max} - \delta_q \quad (37)$$

All strings that do not verify conditions (36) and (37) must be discarded. For the practical implementation of the list construction, we have two arrays, P and Q, that contain the two levels of excitation, p and q, of each string, σ, belonging to the list, $\mathscr{L}[p_{max}, q_{max}]$.

Test Results and Applications

We performed some FCI calculations on the Ne atom in a $[5s3p2d]$ AO basis with eight active electrons and reported various timings, obtained on a IBM RISC 6000 using the function `MCLOCK()`. The basis set used in our computation is similar to the one used in Refs. 5 and 20, differing only in the fact that we used six-component d-functions and discarded the two highest SCF orbitals of s-symmetry to have exactly the same number of orbitals as a $[5s3p2d]$ basis with five-component d orbitals. In Table III the CPU time for one CI itera-

TABLE III.
Timings Obtained on an IBM RISC 6000 Using the Function `MCLOCK()` for an FCI Calculation on an Ne Atom in a [5*s*3*p*2*d*] AO Basis, with Eight Active Electrons.[a]

Method	CI Iteration	Monoelectronic Lists	Bielectronic Lists
One step	1536	108	62
Two steps	1419	52	8

[a] The time (seconds) for one iteration, for the generation of mono- and bielectronic lists, is reported for both methods described.

tion is compared to the time for the mono- and bielectronic-list generation in the case of one-and two-step methods. We notice that the total time for the list generation (both mono- and bielectronic lists) represents about the 11% of the iteration time with the one-step method, whereas in the case of the two-step method it is about 4%. The two-step method is faster, but, in both cases, the time for the list generation represents only a small fraction of the total time. The direct list generation thus seems more convenient than the list precalculation. If we precalculate the lists, we save 4% or 11% of the CPU iteration time, but the executable size is 54 Mbytes instead of 39 Mbytes, and we have to store on memory or on disk the arrays containing the list addresses and parity factors.

To compare the performance of the one-step and two-step methods, we did some FCI calculations in which the number of electrons was two, three and four, respectively, and the number of orbitals varied from 16 to 64 with a step of eight. In each case we determined the time spent for the generation of mono- and bielectronic lists with the two methods. The results for the monoelectronic lists are reported in Tables IV and V, and those for the bielectronic ones in Tables VI and VII. In each table we report the number of examined and selected strings (n_{ex}, n_{sel}), the total time spent (T) and the number of examined and selected strings in the time unit (v_{ex}, v_{sel}).

We first consider the monoelectronic-list generation. From Tables IV and V we notice that T decreases slightly in going from the one-step to the two-step method. In the cases with two electrons the reduction goes from a factor of two (32 orbitals) to a factor of 7 (64 orbitals). In the three-electron cases it goes from a factor of two (24 orbitals) to a factor of 15 (64 orbitals). The four-

TABLE IV.
Monoelectronic List Generation with the One-Step Method.[a]

n	*m*	n_{sel}	n_{ex}	T	v_{sel}	v_{ex}
16	2	49,456	276,480	0.22	0.1339	1.2567
24	2	230,824	3,059,184	1.67	0.1382	1.8318
32	2	989,280	17,006,848	7.88	0.1255	2.1582
40	2	3,051,400	64,552,800	39.37	0.0775	1.6396
48	2	7,648,880	192,229,248	109.28	0.0700	1.7591
56	2	16,621,416	483,935,760	261.37	0.0636	1.8515
64	2	32,539,584	1,077,156,864	560.79	0.0580	1.9208
16	3	191,464	1,290,240	0.78	0.2454	1.6542
24	3	2,423,652	22,434,016	11.87	0.2042	1.8900
32	3	14,344,560	170,068,480	85.09	0.1686	1.9987
40	3	56,450,900	817,668,800	531.62	0.1062	1.5381
48	3	172,099,800	2,947,515,136	1,902.52	0.0946	1.5493
56	3	440,467,524	8,710,843,680	5,569.28	0.0788	1.5641
64	3	992,457,312	22,261,241,856	13,906.30	0.0714	1.6008
16	4	765,856	4,193,280	2.82	0.2716	1.4870
24	4	16,157,680	117,778,584	75.22	0.2148	1.5658
32	4	133,882,560	1,232,996,480	821.96	0.1629	1.5001
40	4	677,410,800	7,563,436,400	6,201.50	0.1092	1.2196
48	4	2,524,130,400	33,159,545,280	26,068.55	0.0968	1.2720

[a] For each case we report the number of orbitals (n), electrons (m), selected (n_{sel}) and examined (n_{ex}) strings, time spent (T) in seconds, and number of selected and examined strings in the time unit ($v_{sel} = n_{sel}$, $T * 10^{-6}$ and $v_{ex} = n_{ex} / T * 10^{-6}$).

TABLE V.
Monoelectronic List Generation with the Two-Step Method.[a]

n	m	n_{sel}	n_{ex}	T	v_{sel}	v_{ex}
16	2	29,456	34,560	0.92	0.0320	0.0376
24	2	230,824	254,932	1.76	0.1312	0.1449
32	2	989,280	1,062,928	3.78	0.2617	0.2812
40	2	3,051,400	3,227,640	9.43	0.3236	0.3423
48	2	7,648,880	8,009,552	20.43	0.3744	0.3920
56	2	16,621,416	17,283,420	42.29	0.3930	0.4087
64	2	32,539,584	33,661,152	77.61	0.4193	0.4337
16	3	191,464	241,920	1.28	0.1496	0.1890
24	3	2,423,652	2,804,252	5.23	0.4634	0.5362
32	3	14,344,560	15,943,920	25.28	0.5674	0.6307
40	3	56,450,900	61,325,160	97.06	0.5816	0.6318
48	3	172,099,800	184,219,696	303.22	0.5676	0.6075
56	3	440,467,524	466,652,340	816.21	0.5397	0.5717
64	3	992,457,312	1,043,495,712	943.47	0.5107	0.5369
16	4	765,856	1,048,320	2.50	0.3063	0.4193
24	4	16,157,680	19,629,764	31.57	0.5118	0.6218
32	4	133,882,560	154,124,560	264.63	0.5059	0.5824
40	4	677,410,800	756,343,640	1,371.71	0.4938	0.5514
48	4	2,524,130,400	2,763,295,440	5,232.59	0.4824	0.5280

[a] For each case we report the number of orbitals (n), electrons (m), selected (n_{sel}) and examined (n_{ex}) strings, time spent (T) in seconds, and number of selected and examined strings in the time unit ($v_{sel} = n_{sel}/T * 10^{-6}$ and $v_{ex} = n_{ex}/T * 10^{-6}$).

electron calculations were performed only up to 48 orbitals, because the calculations with 56 or 64 orbitals would have implied very long timings, especially with the one-step method. In the cases reported the time reduction goes from two (24 orbitals) to five (48 orbitals). In Figure 1 the timings for the monoelectronic lists with the two methods are reported. From the comparison of one- and two-step curves in the different cases (two, three, four electrons) we observe that the two-step curve essentially lies below the corresponding one-step curve, but the two curves in each case are almost parallel, which means that there is a reasonable advantage with the two-step method compared to the one-step method.

In the case of the bielectronic lists, on the contrary, the two-step method is extremely convenient compared to the one-step method. From Tables VI and VII, we observe that, in the case of two electrons in 16 orbitals, the CPU time is almost the same using the two methods, whereas with 24 orbitals the time is reduced by a factor of 7, and the time reduction increases drastically with the number of orbitals: in the case of two electrons in 64 orbitals the time is reduced by a factor of 80. In the three-electron cases, the time reduction goes from a factor of two (16 orbitals) to a factor of 70 (64 orbitals). In the four-electron cases the time reduction goes from a factor of two (16 orbitals) to a factor of 24 (48 orbitals). In Figure 2 the bielectronic-list timings are reported. We notice that the two-step curve lies below the corresponding one-step curve and the gap between two corresponding curves increases with the number of orbitals much more than for the monoelectronic lists. Thus, we can say that the two-step is more convenient than the one-step method for the generation of monoelectronic lists, and it is extremely convenient for the bielectronic lists. For a fixed number of orbitals, the time saving is larger when a small number of electrons is used; that is, with two electrons in 48 orbitals, the two-step method is more convenient compared to the one-step, than with three or four electrons in the same number of orbitals.

The fact that the time reduction is smaller in the monoelectronic than in the corresponding bielectronic cases can be explained if we consider how

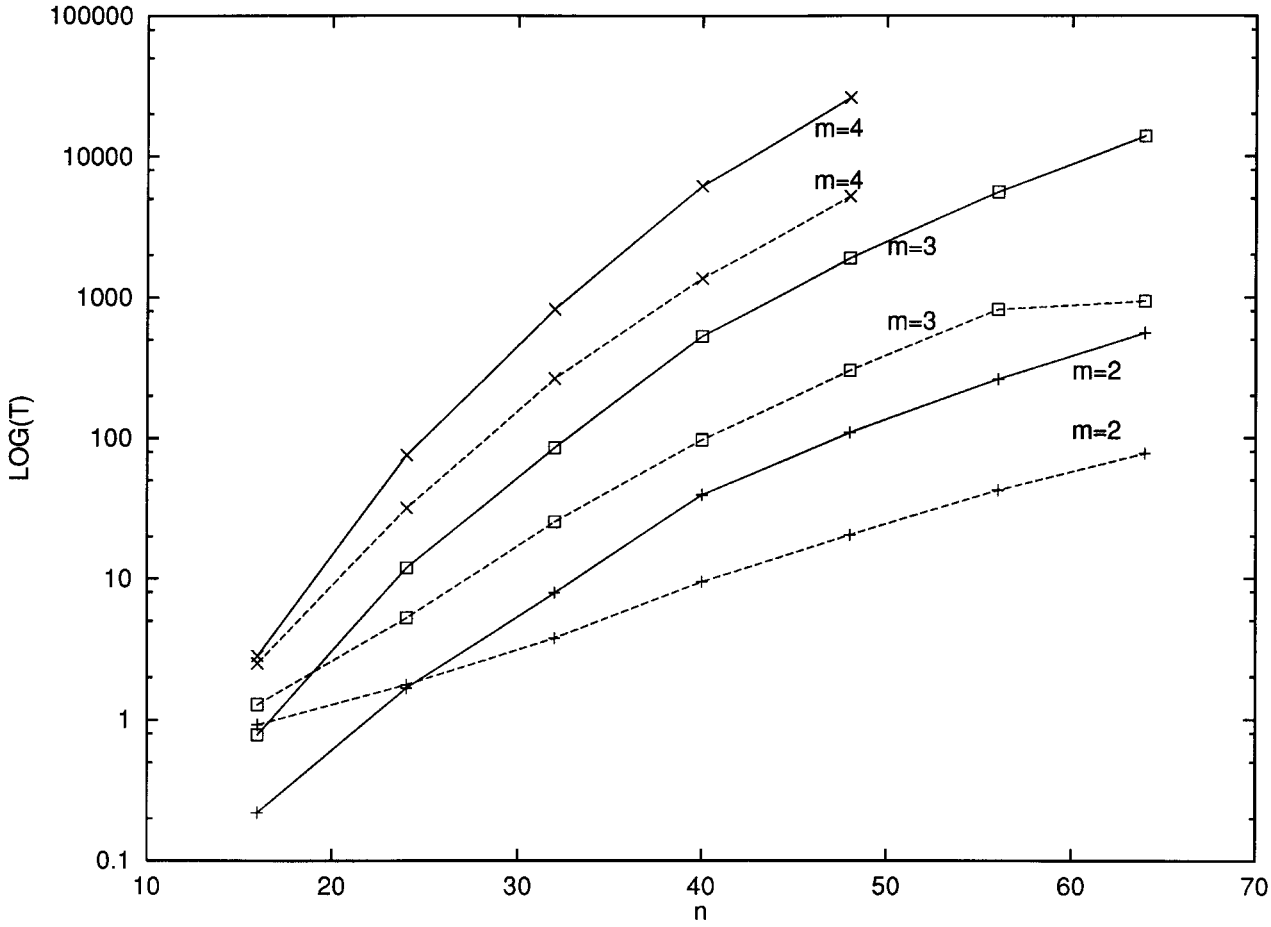

FIGURE 1. Timing, (T), in seconds, for the monoelectronic-list generation: T as a function of the number, n, of orbitals for the one-step (dashed lines), and two-step (full lines) methods. The number, m, of electrons is indicated on each curve.

the lists are generated. For the bielectronic cases, we have to build the preliminary list of type L_{ij}^{ij} with i and j occupied and then choose among the strings of this list those with k and l empty, to form the final list L_{ij}^{kl}. In the monoelectronic case, we first produce the list L_i^i with i occupied, and among the strings of this list, we choose those with k virtual to form the list L_i^k. In this case, the preliminary list, L_i^i, is much longer than in the bielectronic case, and more time is thus required to examine all the strings of L_i^i than those of the preliminary list L_{ij}^{ij}. If we compare the results with the two-step method for mono- and bielectronic lists (Tables V and VII), we notice that, in each case, the number of both the examined strings (n_{ex}) and selected strings (n_{sel}) is larger for the monoelectronic lists than for the bielectronic ones, and, as a result, the time spent in monoelectronic-list generation is longer than in bielectronic-list generation.

In regard to the velocities, we first consider the results obtained with the one-step method. For the monoelectronic lists (Table IV), the number of examined strings per second, v_{ex}, is almost constant for the cases with the same number of electrons, whereas the number of selected strings per second, v_{sel}, decreases within each group of cases with the same number of electrons. We also observe that v_{ex} is larger than v_{sel}, because the number of the examined strings is much larger than of the selected ones. In the bielectronic case (Table VI), both v_{ex} and v_{sel} are almost constant within each group, and v_{ex} is larger than v_{sel} for the same reason as above. If we consider the results with the two-step method, we observe that, in the monoelectronic case (Table VII), v_{ex} and v_{sel} increase

TABLE VI.
Bielectronic List Generation with the One-Step Method.[a]

n	m	n_{sel}	n_{ex}	T	v_{sel}	v_{ex}
16	2	1856	222,720	0.14	0.0133	1.5909
24	2	9648	2,662,848	1.20	0.0080	2.2190
32	2	30,976	15,364,096	5.58	0.0056	2.7534
40	2	76,400	59,592,000	26.25	0.0029	2.2702
48	2	159,552	179,974,656	78.55	0.0020	2.2912
56	2	297,136	457,589,440	188.18	0.0016	2.4317
64	2	508,928	1,025,998,848	406.75	0.0013	2.5224
16	3	22,736	1,039,360	0.47	0.0484	2.2114
24	3	194,568	19,527,552	6.52	0.0298	2.9950
32	3	871,200	153,640,960	47.21	0.0185	3,2544
40	3	2,758,040	754,832,000	307.78	0.0090	2,4525
48	3	7,033,584	2,759,611,392	1,141.74	0.0062	2.4170
56	3	15,472,296	8,236,609,920	3,274.78	0.0047	2.5151
64	3	30,567,488	21,203,976,192	8,545.18	0.0036	2.4814
16	4	128,184	3,377,920	14.0	0.0916	2.4128
24	4	1,865,556	102,519,648	33.85	0.0551	3.0286
32	4	11,817,840	1,113,896,960	358.02	0.0330	3.1112
40	4	48,407,820	6,982,196,000	3,038.64	0.0159	2.2978
48	4	151,522,344	31,045,628,160	13,925.12	0.0109	2.2295

[a] For each case we report the number of orbitals (n), electrons (m), selected (n_{sel}) and examined (n_{ex}) strings, time spent (T) in seconds, and number of selected and examined strings in the time unit ($v_{sel} = n_{sel}/T * 10^{-6}$ and $v_{ex} = n_{ex}/T * 10^{-6}$).

TABLE VII.
Bielectronic List Generation with the Two-Step Method.[a]

n	m	n_{sel}	n_{ex}	T	v_{sel}	v_{ex}
16	2	1856	1856	0.15	0.0124	0.0124
24	2	9648	9648	0.14	0.0689	0.0689
32	2	30,976	30,976	0.35	0.0885	0.0885
40	2	76,400	76,400	0.85	0.0899	0.0899
48	2	159,552	159,552	1.73	0.0922	0.0922
56	2	297,136	297,136	3.05	0.0974	0.0974
64	2	508,928	508,928	4.92	0.1034	0.1034
16	3	22,736	25,872	0.28	0.0812	0.0924
24	3	194,568	212,256	1.35	0.1441	0.1572
32	3	871,200	929,280	4.79	0.1819	0.1940
40	3	2,758,040	2,903,200	13.84	0.1993	0.2098
48	3	7,033,584	7,339,392	33.47	0.2101	0.2193
56	3	15,472,296	16,045,344	69.73	0.2219	0.2301
64	3	30,567,488	31,553,536	125.79	0.2430	0.2508
16	4	128,184	168,896	0.68	0.1885	0.2484
24	4	1,865,556	2,228,688	7.21	0.2587	0.3091
32	4	11,817,840	13,474,560	41.64	0.2838	0.3236
40	4	48,407,820	53,709,200	187.39	0.2714	0.3011
48	4	151,522,344	165,136,320	552.99	0.2740	0.2986

[a] For each case we report the number of orbitals (n), electrons (m), selected (n_{sel}), and examined (n_{ex}) strings, the time spent (T) in seconds, and the number of selected and examined strings in the time unit ($v_{sel} = n_{sel}/T * 10^{-6}$ and $v_{ex} = n_{ex}/T * 10^{-6}$).

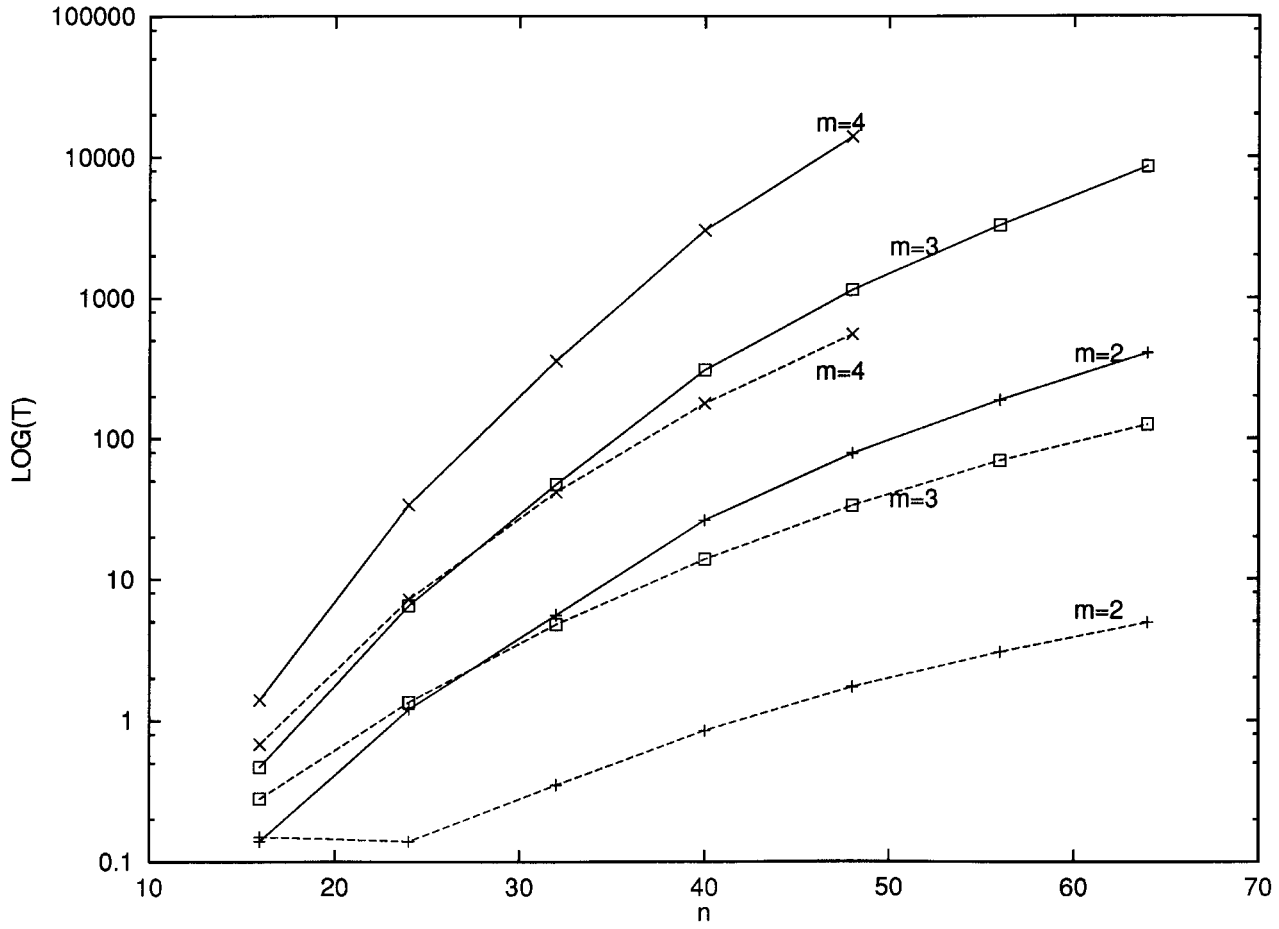

FIGURE 2. Timing, (T), in seconds, for the bielectronic-list generation: T as a function of the number, n, of orbitals for the one-step (dashed lines), and two-step (full lines) methods. The number, m, of electrons is indicated on each curve.

within each group of cases with the same number of electrons and they are almost equal because the numbers of examined and selected strings are nearly the same. In the bielectronic two-step case (Table VII), v_{ex} is almost equal to v_{sel}, and both increase within each group. In the cases with two electrons, v_{ex} and v_{sel} coincide exactly because all of the strings of the preliminary partial list L_{ij}^{ij} belong also to the final list L_{ij}^{kl}, because there are only two electrons. We also observe that, within the two-electron cases, the velocity increase from 16 to 64 orbitals is larger than in the corresponding three- and four-electron cases.

The direct-list algorithm is now part of our FCI/TCI code,[12,21] although its benefit is relevant only for relatively large basis sets. In particular, we used this algorithm to study, at FCI and TCI level, the BH molecule in its ground state, $X^1\Sigma^+$, and in one of its excited states, $A^1\Pi$. We used different ANO-type basis sets of increasing quality, up to a $[6s5p3d2f1g]$ basis set on boron and $[4s3p2d1f]$ on the hydrogen atom. In the FCI calculations only the four valence electrons were correlated, whereas at the TCI level the correlation of the core electrons was also included. The information regarding the whole study on the BH molecule will be presented in a forthcoming publication.[22] The truncation procedure described previously allowed us to include a large number of orbitals in the six-electron calculations. For the first two TCI calculations the corresponding FCI ones with six correlated electrons were also performed, and the results obtained within the two methods differed by about 0.01%, which indicates that the truncation scheme is a good approximation of the FCI. By using direct-list generation it was possible to perform some of the calculations on an IBM RISC 6000 with 128 megawords of core memory,

whereas the indirect scheme would have implied the use of a larger workstation or even a supercomputer.

Another application was the study of the HeH^- ground-state energy curve at the FCI level.[23] The attractive well of this four-electron system is only a few cm^{-1} deep, and large basis set are to be used, even to obtain qualitatively correct results. The problem is made more difficult by the presence of the H^- ion, whose description requires the use of very diffuse orbitals. We performed FCI calculation on this system using a $[9s4p3d]$ basis set for helium, and $[12s8p5d]$ for hydrogen. This comprises a total of 97 orbitals, and an FCI space of 5,735,520 symmetry-adapted determinants using the C_{2v} group. Although the size of the FCI space is not very large, the calculation is rather heavy because of the large number of orbitals. The presence of only four electrons implies that the two-electron lists are extremely short, and therefore our direct-list FCI algorithm is particularly convenient. With our original algorithm, the code requires at least 282.5 megawords of core memory to be executed, and hence the use of a large workstation or a supercomputer. The direct-list algorithm dramatically reduces the memory requirement, and 97.4 megawords are sufficient. The whole calculation (about 20 points on the energy curve) was done on a relatively small workstation, with 128 megawords of core memory.

Discussion and Conclusions

To have an idea of the general features of our algorithm, we now discuss the timings. The operation count of $H_{\alpha\beta}$ is larger than that of $H_{\beta\beta}$,[12] whereas, on the other hand, the length of the bielectronic lists by far exceeds that of the monoelectronic lists. Thus, from our point of view, the $\beta\beta$ case is probably more important and we shall consider it first. In this case the timings obtained (see Tables VI and VII) correspond roughly to the following velocities:

$$v_{ex} \simeq 2.0\ M_{op}/\text{second} \tag{38}$$

$$v_{sel} \simeq 0.2\ M_{op}/\text{second} \tag{39}$$

The CI multiplication was done with a SAXPY routine, which, on the computer used, has a velocity of the order of:

$$v_{saxpy} \simeq 10.0\ M_{oP}/\text{second} \tag{40}$$

(With M_{op}, i.e., million of operations, we indicate the number of list elements examined, or of elementary multiplications in the SAXPY operation.) Given these velocities, we can say that for each quadruple i, j, k, l, the total time is given by:

$$T \simeq L/v_{ex} + l/v_{sel} + l\mathcal{N}/v_{saxpy} \tag{41}$$

where L, l, and $\mathcal{N}$ indicate the length of the examined list, the selected list, and the total number of strings, respectively. Inserting eqs. (38), (39), (40) into eq. (41), T becomes:

$$T \simeq 0.5L + 5l + l\mathcal{N}0.1 \tag{42}$$

For the direct-list generation to be efficient, it is necessary that the two first terms of the right-hand side of eq. (42) are negligible with respect to the third one. If, for example, the time for the lists does not exceed 10% of the multiplication time, the range of $\mathcal{N}$ is given by:

$$l\mathcal{N}0.1 > 10(0.5L + 5l) \tag{43}$$

In the one-step case we have, in general, $l \ll L$ and $L = \mathcal{N}$, and from eq. (42), it turns out that:

$$T \simeq 0.5L + 0.1Ll \tag{44}$$

If we now use condition (43), we notice that the one-step method is convenient when $l > 50$, and this accounts for the fact that, in the cases with a few electrons, where l is small, the one-step-method is not suitable. In the two-step case, on the other hand, $L \simeq l$, and T becomes:

$$T \simeq 5.5l + 0.1\mathcal{N}l \tag{45}$$

From condition (43) it turns out that the two-step method becomes appropriate when $\mathcal{N} > 550$; that is, in any case of practical interest. It should be emphasized that, because in $a_i^+ a_j^+ a_k a_l$ we can interchange index i with j, and k with l when all the spins are equal, the total number of quadruples i, j, k, l to be considered is reduced by a factor of $1/4$.

Similar to the $\beta\beta$ case, in the $\alpha\beta$ case, for each quadruple i, j, k, l, the total time is given by:

$$T \simeq L/v_{ex} + l/v_{sel} + l^2/v_{saxpy} \tag{46}$$

where L, l, and $\mathcal{N}$, as before, indicate the length of the list examined, the selected list, and the total number of strings, respectively. All the quadruples have to be considered in this case, and the operation count is thus four times that of the $\beta\beta$ case. In the $\alpha\beta$ case the analysis is less simple. Analo-

gously to eq. (43) we obtain:

$$l^2/v_{saxpy} > 10(L/v_{ex} + l/v_{sel}) \qquad (47)$$

(We did not take into account the times related to the *gather* and *scatter* operations, which are used in the actual implementation of the $\alpha\beta$ part.) An explicit computation of the function:

$$F(n, m) = l^2/v_{saxpy} - 10(L/v_{ex} + l/v_{sel}) \quad (48)$$

for $v_{ex} \simeq 1.5\ M_{op}$/second (see Table IV), $v_{sel} \simeq 0.5\ M_{op}$/second (see Table V), and $v_{saxpy} \simeq 10.0\ M_{op}$/second, and shows that the one step method is convenient except for $m = 2$, $m = 3$, whereas the two step is always convenient, except for $m = 2$ and very small n.

To conclude, we have presented two variants of a method for the direct generation of string lists in a CI algorithm. The direct list generation is particularly convenient whenever the one- and two-electron lists are too long to be held in the core memory. A limit case of this type is obtained by working in the AO basis instead of the MO one. Thus, in principle, one could consider the extension of integral-driven direct CI to the AO basis. Unfortunately, the practical implementation of such an algorithm would require the solution of difficult problems connected with the nonorthogonality of the orbital basis, and much work in this area is needed. Indeed, this is at present the main drawback preventing the implementation of efficient large-scale valence-bond codes.

The direct generation of the lists has several advantages compared to the list precalculation, as a consequence of the reduced memory requirements. The first version, called the one-step method, is particularly suitable when the number, m, of electrons per string is ≥ 4, whereas the second one (two-step method) is convenient for $m < 4$ also. The method developed has been be used to perform FCI calculations including a large number of orbitals in the basis set.[22,23] In this way the FCI programs will not only be tools to assess the reliability of approximate methods like truncated CI, MCSCF, or coupled cluster, but will be helpful in understanding the electronic structure of molecules in some difficult cases, where the usual approximations are not sufficient.

References

1. N. C. Handy, *Chem. Phys. Lett.*, **74**, 280 (1980).
2. P. Saxe, H. F. Schaefer III, and N. C. Handy, *Chem. Phys. Lett.*, **79**, 202 (1981).
3. P. E. M. Siegbahn, *Chem. Phys. Lett.*, **109**, 417 (1984).
4. P. J. Knowles and N. C. Handy, *Chem. Phys. Lett.*, **111**, 315 (1984).
5. J. Olsen, B. O. Roos, P. Joergensen, and H. J. Å. Jensen, *J. Chem. Phys.*, **89**, 2185 (1988).
6. P. J. Knowles, *Chem. Phys. Lett.*, **155**, 513 (1989).
7. P. J. Knowles and N. C. Handy, *J. Chem. Phys.*, **91**, 2396 (1989).
8. P. J. Knowles and N. C. Handy, *Comput. Phys. Commun.*, **54**, 75 (1989).
9. S. Zarrabian, C. R. Sarma, and J. Paldus, *Chem. Phys. Lett.*, **155**, 183 (1989).
10. R. J. Harrison and S. Zarrabian, *Chem. Phys. Lett.*, **158**, 393 (1989).
11. J. Olsen, P. Jørgensen, and J. Simons, *Chem. Phys. Lett.*, **169**, 463 (1990).
12. G. L. Bendazzoli and S. Evangelisti, *J. Chem. Phys.*, **98**, 3141 (1993).
13. R. Ansaloni, E. Rossi, and S. Evangelisti, *Lecture Notes in Computer Science*, Vol. *919*, Springer, Berlin, 1995.
14. S. Evangelisti, G. L. Bendazzoli, R. Ansaloni, F. Duri, and E. Rossi, *Chem. Phys. Lett.*, **252**, 437 (1996).
15. C. Lanczos, *J. Res. Nat. Bur. Stand.*, **45**, 255 (1950).
16. E. R. Davidson, *J. Comp. Phys.*, **17**, 87 (1975).
17. B. O. Roos, *Chem. Phys. Lett.*, **15**, 153 (1972).
18. B. O. Roos and P. E. M. Siegbahn, in *Modern Theoretical Chemistry*, *Vol. 3*, H. F. Schaefer III, Ed., Plenum Press, New York, 1977.
19. D. H. Lehmer, in *Applied Combinatorial Mathematics*, E. F. Beckenbach, Ed. Wiley, New York (1964); see also W. Duch, *J. Phys. A: Math. Gen.*, **18**, 3283 (1985).
20. C. W. Bauschlicher, Jr., S. R. Langhoff, P. R. Taylor, and H. Partridge, *Chem. Phys. Lett.*, **126**, 436 (1986).
21. G. L. Bendazzoli and S. Evangelisti, *Gazz. Chim. Ital.*, **126**, 619 (1996).
22. L. Gagliardi, G. L. Bendazzoli, and S. Evangelisti, *Mol. Phys.* (in press).
23. G. L. Bendazzoli, S. Evangelist, and F. Passarini, *Chem. Phys.*, **215**, 217 (1997).